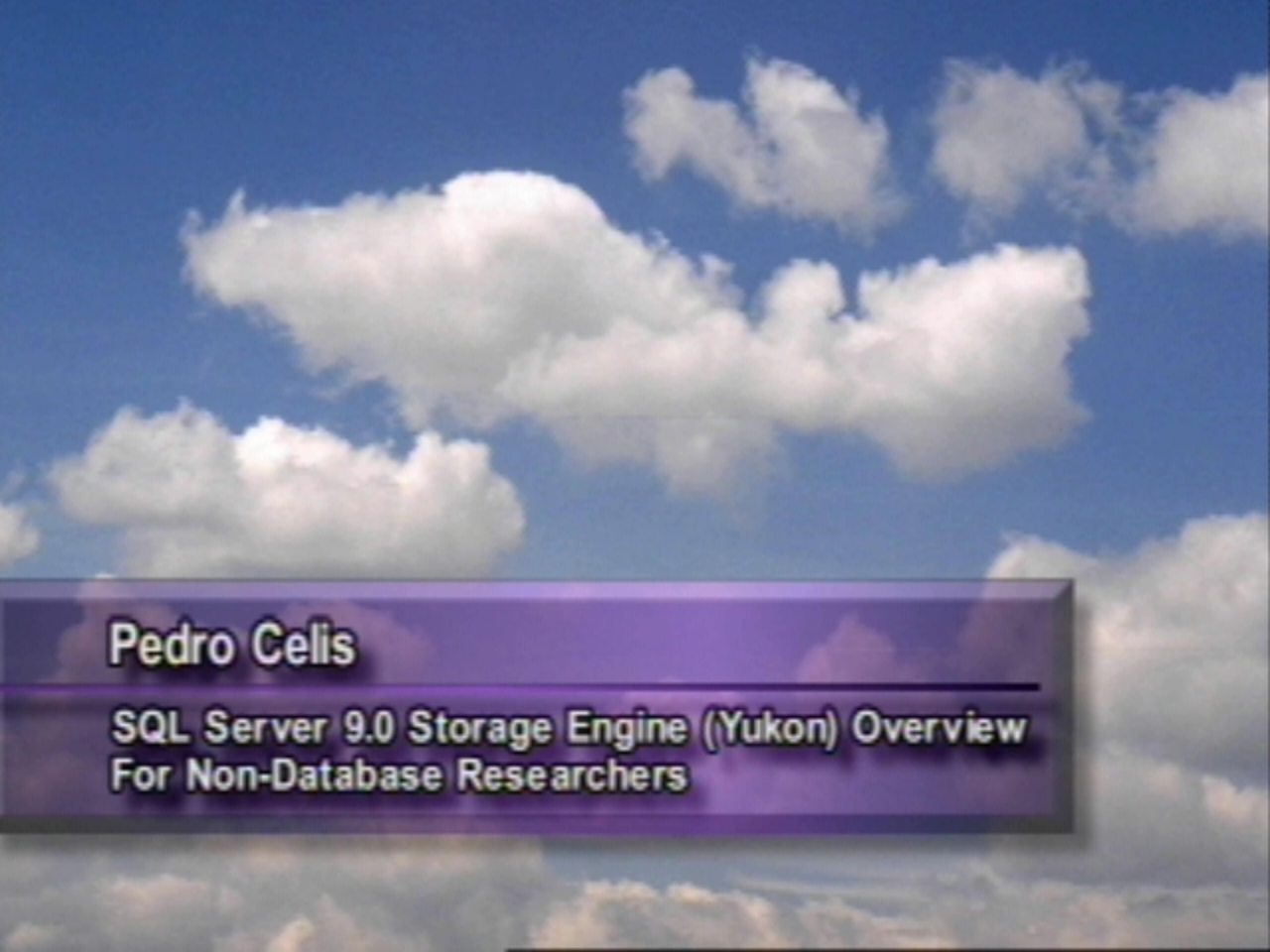

Pedro Celis
SQL Server 9.0 Storage Engine (Yukon) Overview
For Non-Database Researchers

Presentation Goals

- Describe where we've come from
- Overview of Yukon scope
- Some technical details on Yukon
- Solicit feedback
- Initiate ongoing dialog

Where we've come from

- SQL 7.0
  - We mostly knew what we were going to build.
  - Needed to make sure what we built worked:
    - New QP, Parallelism, new storage engine

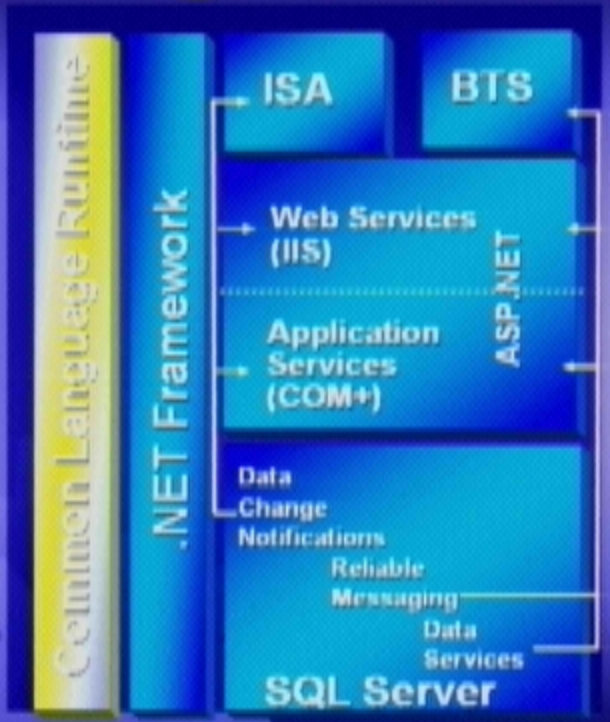
Where we've come from

- SQL 7.0
  - We mostly knew what we were going to build.
  - Needed to make sure what we built worked:
    - New QP, Parallelism, new storage engine
- SQL 2000
  - Clean up and several new features
  - (We thought we'd need a new release quickly - we were wrong...)

Where we've come from

- SQL 7.0
  - We mostly knew what we were going to build.
  - Needed to make sure what we built worked:
    - New QP, Parallelism, new storage engine
- SQL 2000
  - Clean up and several new features
  - (We thought we'd need a new release quickly - we were wrong...)
- Yukon
  - Innovation!
  - Continued evolution of architecture established in 7.0

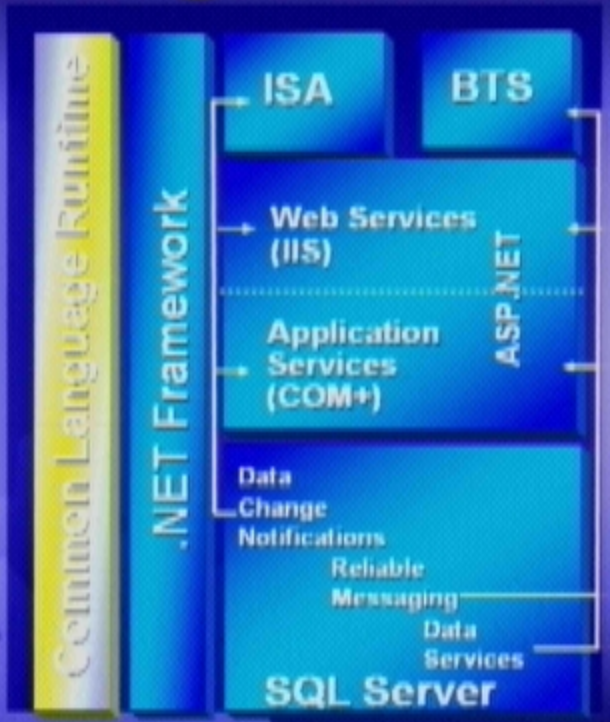
Winning with .NET Servers



"Better Together"

- CLR
  - Language Freedom
- XML/SOAP
  - Open protocol
- Caching
  - All Levels
- Diagnostics
  - All Levels
- Security
  - Unifying concepts
- Management
  - SLA focus

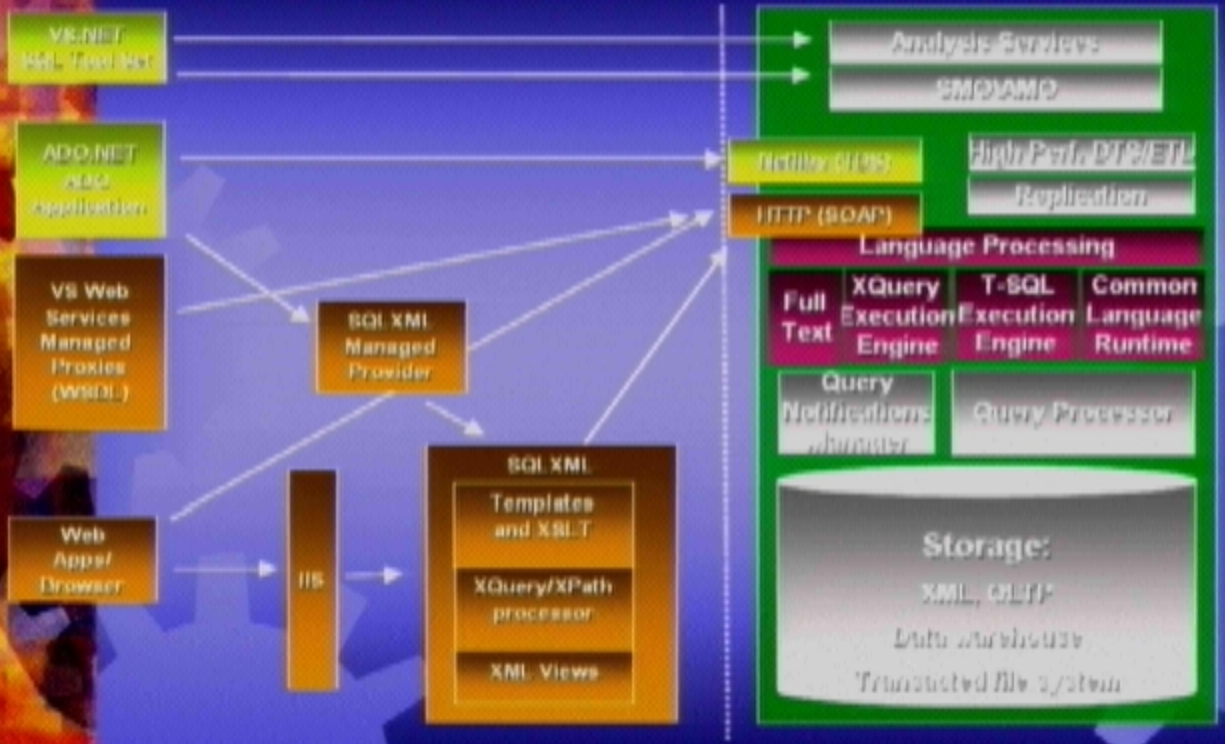
Winning with .NET Servers



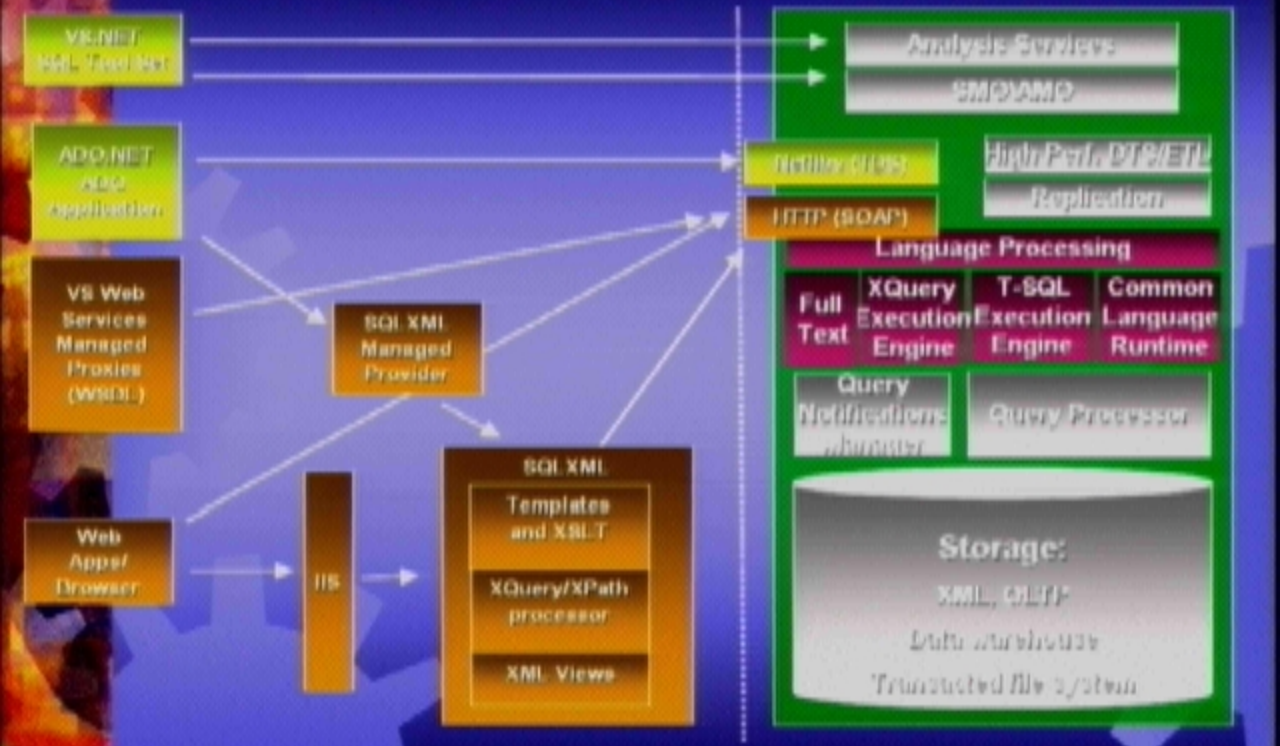
"Better Together"

- CLR
  - Language Freedom
- XML/SOAP
  - Open protocol
- Caching
  - All Levels
- Diagnostics
  - All Levels
- Security
  - Unifying concepts
- Management
  - SLA focus

SQL Server "Yukon"



SQL Server "Yukon"



Yukon Server Investment

- Scalability - Big scale up investment
  - Partitions
  - Online index build
  - SMP enhancements
- Availability - Many new features
  - Online partial database restore
  - Real-time log shipping

Yukon Server Investment

- Programmability
  - CLR Integration
  - XML Integration
  - SQL Enhancements
  - Support for distributed app. Development
  - Versioning

Caveat

- We've just finishing M3 and everything's about 80% done – (Read: some things may still get cut).
- If there's something missing, let us know.
- If you feel passionate about something we are or should be doing, get involved.

Yukon Server Features

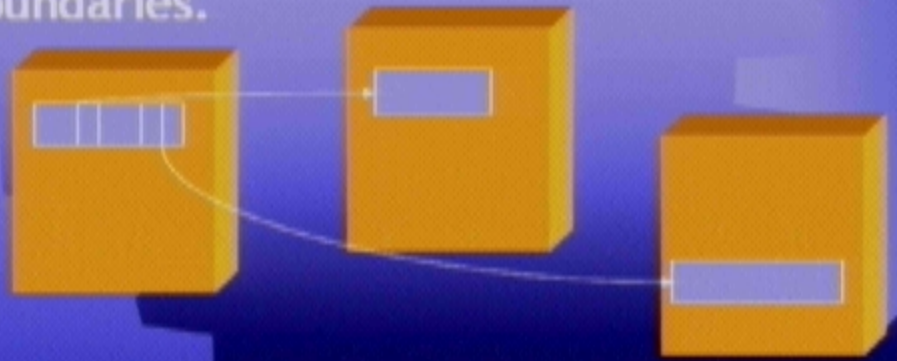
- Access Methods
- Availability Features
- Utilities
- Query Engine Enhancements
- Other stuff

Access Methods

- One of the few major component the Storage Engine team didn't rearchitect for SQL 7
- Completely rearchitected for Yukon
- Support for many new features
  - Large Row Support
  - Merged Indexes
  - Prefix/Suffix Compression
  - Prefix unique indexes

Access Methods

- Large Record Support
  - Use In-Row Text Infrastructure to support large varchar's. (Known as "varchar (max)")
  - Rows > page split on varchar column boundaries.



Access Methods

- Merged Indexes
  - Rows from multiple indexes are stored in the same B-Tree.
  - Yields single I/O behavior for complex relationships
  - Great performance feature for large and small systems

Prefix/Suffix Truncation

- Removes common prefixes on indexes
- Increases storage and scan density
  - More stuff on fewer pages
- Suffix truncation in interior nodes
  - Interior nodes store separators only rather than full index key
  - Results in better density in interior nodes
  - Fewer levels and better density

Prefix Unique Indexes

- Create index on T1 (C1, C2, C3, C4) that's unique on C1, C2 alone
  - Allows unique indexes to "carry" additional information for covering queries
  - Can result in fewer indexes.

Prefix/Suffix Truncation

- Removes common prefixes on indexes
- Increases storage and scan density
  - More stuff on fewer pages
- Suffix truncation in interior nodes
  - Interior nodes store separators only rather than full index key
  - Results in better density in interior nodes
  - Fewer levels and better density

Prefix Unique Indexes

- Create index on T1 (C1, C2, C3, C4) that's unique on C1, C2 alone
  - Allows unique indexes to "carry" additional information for covering queries
  - Can result in fewer indexes.

Versioning

- Alternative to standard Xact Isolation levels
  - (We're retaining the existing isolation levels)
- Provides transaction consistent view of data without having writers block readers or readers block writers.
- Trades lock waits for CPU cycles
- Minimizes deadlocks

Versioning

- Alternative to standard Xact Isolation levels
  - (We're retaining the existing isolation levels)
- Provides transaction consistent view of data without having writers block readers or readers block writers.
- Trades lock waits for CPU cycles
- Minimizes deadlocks

Partition Support

- Intra-node partitioned indexes
  - Create large tables with multiple partitions
  - Manage partitions separately
    - Create/Drop/Index/Restore!
  - Useful for rolling periodic data
    - e.g. Keep 4 previous quarters of data online
- Will support partition level restore.
  - Restore damaged partition with rest of DB online.
- Design point:
  - Partitions ~1TB in size
  - DB supports 10's to 100's of partitions

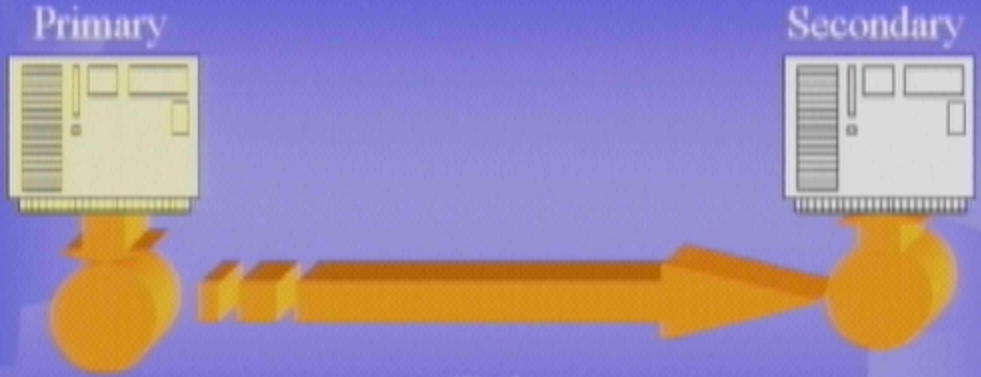
Partition Support

- Intra-node partitioned indexes
  - Create large tables with multiple partitions
  - Manage partitions separately
    - Create/Drop/Index/Restore!
  - Useful for rolling periodic data
    - e.g. Keep 4 previous quarters of data online
- Will support partition level restore.
  - Restore damaged partition with rest of DB online.
- Design point:
  - Partitions ~1TB in size
  - DB supports 10's to 100's of partitions

Backup/Restore

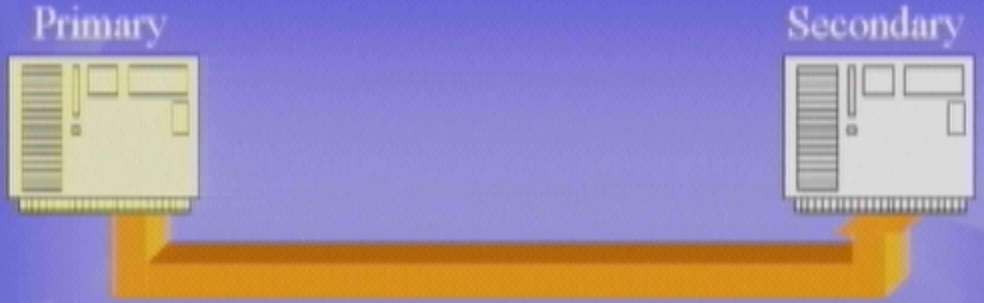
- Mirrored Backups
  - Write mirrored copies to tape to increase reliability
- Checksums on Backups
  - Checksums on data pages and in backup for end-to-end integrity check.
- Online File and Page recovery
  - Recover a file or even a page in a database while the database is online!
- Continue restore on error
  - Let's you fix things up at the end.

Online log shipping



- Current Solution is based upon "backup/copy/restore"
- Secondary lags by copy frequency

Online log shipping



Yukon

- Log records continuously shipped.
- Can control 1 or 2 "safe" copies. (Primary can be set up to wait for secondary write before completing transaction).

Miscellaneous Availability

- Online index build.
  - Create index with read/write activity on table.
- Fast recovery
  - Parallel redo and undo
  - Make database available during undo phase.
- Dedicated Admin Thread
  - Allows SA to get into system and repair a wedged system
- Dynamic AWE memory management

Relational Engine

- Tons of new stuff...
  - CLR
  - XML Data Type
  - Queuing support
  - Query Notifications
  - Statement Level Recompile
  - Fine grained security
  - Query enhancements
  - TSQL Exception Handling

CLR

- Safe, Secure, Scalable, Speedy extensible server programming model
- Easy development of SQL Server apps
  - Same mid- and server-tier programming model
  - Language agnostic
  - UDF's, UDT's, UDA's
- .NET Framework integration
  - .NET languages for stored procedures
    - Min. Goal: performance parity compared to T-SQL
  - Maintain and evolve T-SQL
- Integrated development & debugging in VS .NET

XML Data Type

- XQuery Support
- XML schema support
  - load and store schemas
  - validate and type XML instances
- SQL queries supported over relational and XML data in the server:

```
SELECT id, doc::xquery('for $j in //job return $j/@start,"-",$j/@end')
WHERE doc::exists('//edited[@date > sql:variable("@lastedited")]')
```

Queuing Support

- Native support for queues in server
- Treating tables as queues
  - UPDATE and DELETE with OUTPUTS/ORDER BY/INTO
  - Blocking “read” on SELECT/UPDATE/DELETE (WAITFOR RESULTS)
  - Event notifications on commit/rollback

```
UPDATE T OUTPUTS(inserted.a, deleted.a)
SET T.a = T.a+ 1 WAIT FOR RESULTS 5

SELECT * FROM T WHERE T.x = @1
WAIT FOR RESULTS 10
```

XML Data Type

- XQuery Support
- XML schema support
  - load and store schemas
  - validate and type XML instances
- SQL queries supported over relational and XML data in the server:

```
SELECT id, doc::xquery('for $j in //job return $j/@start,"-",$j/@end')
WHERE doc::exists('//edited[@date > sql:variable("@lastedited")]')
```

Queuing Support

- Native support for queues in server
- Treating tables as queues
  - UPDATE and DELETE with OUTPUTS/ORDER BY/INTO
  - Blocking “read” on SELECT/UPDATE/DELETE (WAITFOR RESULTS)
  - Event notifications on commit/rollback

```
UPDATE T OUTPUTS(inserted.a, deleted.a)
SET T.a = T.a+ 1 WAIT FOR RESULTS 5

SELECT * FROM T WHERE T.x = @1
WAIT FOR RESULTS 10
```

Notification/Eventing

- Query based notification support
- Server notifies client via. SSB queue when result
- Notifications always delivered on server restart
- Events
  - DDL Events (object level)
  - Transaction Events
  - Management Events (Server conditions/health)
  - WMI Bridge

Statement Recompile

- Today, SQL has a batch recompile model
  - Coupling between queries in a batch can result in excessive batch recompiles
  - Yukon defers statement compilation until its required.
  - Two major benefits
    - Avoids coupling issues and wasteful compiles
    - Eliminates compiles on unexecuted branches

Fine Grained Security

- Row level, predicate based security
  - Based upon security filter predicate
- Example:

```
ALTER TABLE Order_Table
{rest of the command here}
ADD CONSTRAINT ConstraintOne
CHECK BEFORE UPDATE ( (IS_MEMBER('MarketingManagers') = 1) OR
( (IS_MEMBER('marketingManagers') = 0) AND 'ShipmentValue' <5000) )
```

T-SQL Exception Handling

- Ability to define handlers for specific exceptions:
- Example:

```
DECLARE HANDLER FOR 8134
  BEGIN
    SET @a = 1
  END

SET @a = 0
SELECT 3 / @a
SELECT 4 / @a
```

Recursive Query Example

- Traverse hierarchy, e.g. bill of materials

```
With Closure(mgrid, empid) as
  Select mgrid, empid from Emp UNION ALL
  Select c.mgrid, e.empid From
    Closure c Join Emp e on c.empid = e.empid

Select * From Closure Where mgrid = 5
```

RE Goodies

- **CUBE w/GROUPING**
  - ANSI syntax for control over specific grouping and rollup dimensions
  - Ensures redundant grouping/rollup not performed
  - Makes our existing cube useful!
- **Data Sampling**
  - Execute query on random data sample
    - Page sampling based on allocation order
  - Supported on tables not queries
  - SELECT * FROM employee e (SAMPLE 10 PERCENT)
- **Recursive Queries**
- **Except/Intersect**

Recursive Query Example

- Traverse hierarchy, e.g. bill of materials

```
With Closure(mgrid, empid) as
  Select mgrid, empid from Emp UNION ALL
  Select c.mgrid, e.empid From
    Closure c Join Emp e on c.empid = e.empid

Select * From Closure Where mgrid = 5
```

Full Text Indexing: Yukon Features

- Performance and Scalability
  - Re-architected
- Integration with SQL Server
  - Full DDL support
  - Backup, restore, & recovery
  - Fulltext queries against views & linked servers
  - Attach/detach, removable DB
  - Replication of FT changes

Full Text Indexing: Yukon Features

- Performance and Scalability
  - Re-architected
- Integration with SQL Server
  - Full DDL support
  - Backup, restore, & recovery
  - Fulltext queries against views & linked servers
  - Attach/detach, removable DB
  - Replication of FT changes

Follow up

- Before we knew what we were going to build and needed your help to make sure we built it right.
- Now, we need your help in building the right thing.
- We'll make a winner - together.

Follow up

- Before we knew what we were going to build and needed your help to make sure we built it right.
- Now, we need your help in building the right thing.
- We'll make a winner - together.